Object Triple Mapping

Bridging Semantic Web and object-oriented programming

The use of Semantic Web and ontologies in creating smart applications has drastically increased during the recent years due to the high potential in allowing machines to reason on provided semantics.

Providing powerful tools that can improve the development of applications based on ontologies can accelerate the adoption of Semantic Web. For this purpose, many tools such as ontology editors, ontology reasoners, triple store frameworks and object triple mapping systems have been developed.

This article introduces the concept of object triple mapping, which essentially consists of a bridge between RDF/OWL and object-oriented programming.

What is object-oriented programming?

Object-oriented programming (OOP) is a paradigm for software development, based on the concept of objects, which can contain data, in the form of fields (often known as *attributes* or *properties*), and code, in the form of procedures (often known as *methods*).

The fundamental characteristics of OOP are:

1. **Inheritance** – Parent to child relationships;
2. **Polymorphism** – Overloading and overriding class members;
3. **Encapsulation** – Hiding data behind operations;
4. **Abstraction** – Reveal mechanisms that are only relevant for the use of other objects.

OOP focuses on the objects that developers want to manipulate rather than the logic required to manipulate them. OOP is well-suited for programs that are large, complex and need constant updates or maintenance. The most popular OOP languages include Java, Python and C++.

In the same way as in the creation of ontologies, in OOP a developer needs to firstly identify all the objects that they want to manipulate as well as the relationships between them (process known as *data modelling*).

What Semantic Web and OOP have in common?

Semantic Web shares a few major characteristics with OOP:

- Semantic Web has **classes** for defining data concepts;
- Semantic Web has **instances** for actual data values;
- **Inheritance** among classes is supported;
- **Strongly typed data types** are supported (string, integer, etc.);
- **Whole/part relationships** are supported. Whole/part relationships are when one class represents the whole object and other classes represent parts. The whole acts as a container for the parts.

Aside from these features, Semantic Web is more rigorous and formal, by specifying extra information such as the allowed relations between classes and their instances.

How OOP and Semantic Web differ?

There are several discrepancies between OOP and RDF/OWL, as defined by the Software Engineering Task Force (SETF):

- Classes in OOP are regarded not as sets to which instances belong, but as types for instances;
- Each instance in OOP belongs to one class as its type's instance;
- Instances in OOP cannot change their type at runtime;
- The list of classes in OOP must be fully known at runtime and it cannot change after that;
- There is no reasoner in OOP that can be used for consistency checking at runtime;
- Properties in OOP are defined locally at the class level and not as stand-alone entities;
- Instances in OOP cannot have arbitrary values for any property without the definition in its class, and no domain constraint.

Challenges for technical experts working with Semantic Web

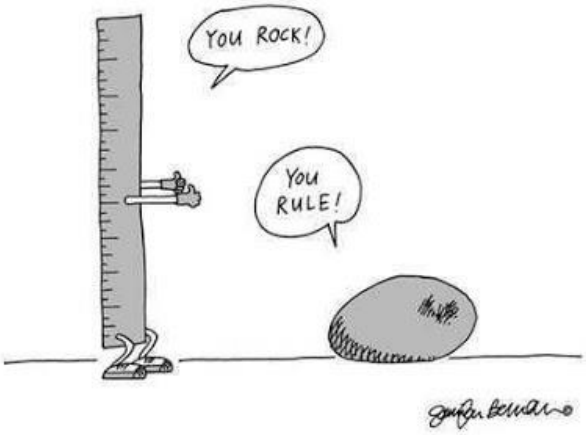
The most challenging concern for a software developer who is trying to model RDF or OWL is to unlearn the OOP acquired practices.

For programmers with OOP background it is quite comfortable to model OWL classes, but they need to unlearn the idea that classes are just static datatype for objects at runtime and start considering classes as dynamic sets of instances that may change membership at any time during runtime.

Database experts are generally more comfortable with the notion of OWL classes as sets, but they still need to resist the appeal of normalizing the data model using primary and foreign keys and focus on modelling objects that accurately represent the information model.

Small example below from JOPA: Accessing Ontologies in an Object-oriented way paper:

"(…), an ontology specifies that each *Person* has a *name*. Due to the open world assumption, the ontology is consistent even if a particular *Person* does not have recorded his/her *name*. However, a genealogical application accessing the ontology needs the *name* to be known, which causes the application to crash whenever it receives (consistent, but application–incompatible) data from an ontological source, specifying a *Person* without a *name*."



Introducing Object Triple Mapping

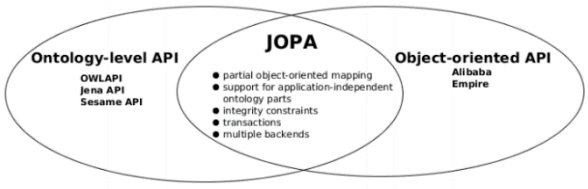
Object triple mapping (OTM) was designed for helping to develop ontology-based applications through the object-oriented programming paradigm.

OTMs map RDF triples into data objects, enabling developers to handle RDF triples as objects in object-oriented applications. In the same manner, developers of relational databases have been using object-relational mapping (ORM) systems (e.g. DQL or Hibernate) to map relational databases into objects.

OTM with JOPA

There are several libraries for multiple programming languages that have been introduced for mapping RDF model to the object model.

One that was particularly tested by the ONE Record team (see code on Github) is JOPA (Java OWL Persistence API).



JOPA and related approaches

JOPA is a domain-specific library which express the domain model in the form of object classes, and their properties (attributes or relationships). JOPA automatically maps the notions from the object-oriented realm to the semantic data model and vice versa.

JOPA compiles integrity constraints into a set of Java annotations that keep their semantics at the object level. This allows easy validation of restrictions such as cardinality, as well as domains and ranges of annotations, data or object properties.

The figure below illustrates a simple Java class generated from an ontology. We can recognize in the annotations the same language used in the creation of ontologies, such as DataProperty, ObjectProperty, etc.

```
@OWLClass(
  iri="http://example.org/Student")
class Student {
  @Id
  URI id;
  @DataProperty(
    iri="http://example.org/name")
  String name;
  @DataProperty(
    iri="http://example.org/email")
  String email;
  @ObjectProperty(
    iri="http://example.org/course")
  Set<Course> courses;
  @Inferred
  @Types
  Set<String> types;
  @Properties
  Map<String, Set<String>> properties;
}
```

Generated Java class using JOPA

Conclusion

Nowadays, a huge amount of existing applications is coded with object-oriented programming languages such as Java or Python.

A key point for accelerating the adoption of Semantic Web is the creation and usage of tools such as object triple mapping frameworks which translate RDF/OWL to object-oriented models. This kind of tools are crucial as they save hours of development and maintenance and translate ontologies in languages that are more familiar to developers. Many OTM libraries exist, however these libraries differ a lot in their capabilities and there is still a lot of work to be done in this area.

More info at https://www.iata.org/one-record/.